

Explicación de las librerías y por qué se utilizan

#include <ESP8266WiFi.h>

- Librería oficial para manejar WiFi específicamente en el ESP8266.
- Se usa en vez de una librería WiFi genérica porque:
 - Está optimizada para este chip.
 - Expone constantes y funciones específicas (como WL_CONNECTED, WiFi.mode(), etc.).
 - Evita problemas de compatibilidad que podrían aparecer usando librerías pensadas para otros microcontroladores (por ejemplo, ESP32).

#include <Firebase_ESP_Client.h>

- Cliente especializado para Firebase en microcontroladores (ESP8266, ESP32).
- Se usa en vez de hacer peticiones HTTP “a mano” porque:
 - Maneja autenticación, reconexión, timeouts y certificados de seguridad internamente.
 - Proporciona métodos de alto nivel (getInt, getString, getJSON, setJSON, etc.).
 - Es más robusto y eficiente en RAM que implementar un cliente HTTP + JSON propio.

#include <Wire.h>

- Librería estándar para comunicación I²C (TWI).
- Necesaria para comunicarse con la pantalla OLED SSD1306.
- Se usa en lugar de “bit banging” con digitalWrite, porque:
 - El I²C nativo es más veloz y estable.
 - Reduce el código de bajo nivel y hace el programa más legible y mantenible.

#include <SPI.h>

- Librería estándar para comunicación SPI.
- Se utiliza para el lector RFID MFRC522, que trabaja sobre bus SPI.
- En lugar de simular SPI manualmente:
 - Permite mayor velocidad de lectura/escritura.
 - Minimiza errores de sincronización y de tiempo.

#include <MFRC522.h>

- Librería específica para el módulo RFID RC522.
- Implementa internamente los comandos de bajo nivel (anticolisión, lectura de UID, etc.).
- Se usa en vez de escribir un driver propio porque:
 - Es una librería probada y ampliamente utilizada.
 - Evita tener que implementar desde cero los protocolos RFID (por ejemplo, ISO14443).

#include <Adafruit_GFX.h> y #include <Adafruit_SSD1306.h>

- Adafruit_GFX: librería genérica de gráficos (texto, líneas, rectángulos, píxeles, etc.).
- Adafruit_SSD1306: driver específico para pantallas OLED con chip SSD1306.
- Se usan en vez de otras alternativas porque:
 - Son muy estables, bien documentadas y populares.
 - Tienen buen soporte para ESP8266.
 - Permiten manejar texto y gráficos sin escribir código de bajo nivel para la pantalla.

#include <NTPClient.h> y #include <WiFiUdp.h>

- NTPClient: cliente sencillo para obtener la hora actual desde servidores NTP.
- WiFiUdp: permite enviar y recibir paquetes UDP sobre WiFi.
- Se usan en lugar de implementar NTP manualmente porque:
 - Abstraen todos los detalles del protocolo NTP (paquetes binarios, puertos, etc.).
 - Permiten obtener hora formateada y en formato epoch de forma simple.
 - Facilitan aplicar el desfase horario (por ejemplo, zona Chile: UTC-3).

#include <time.h>

- Librería estándar de C para manejo de fechas y tiempos (time_t, struct tm, etc.).
- Se utiliza para convertir el tiempo epoch en una fecha legible (formato AAAA-MM-DD).
- Complementa a NTPClient, permitiendo registrar en Firebase tanto la hora como la fecha de forma humana y ordenada